Notes on C/AL[1]

"A Strange Site"

http://astrangesite.altervista.org

powered by

dr. Alessandro Strano

MESSAGE ('Message'[,value1,..])

Shows a message.
Example of composite message

```
('This is written on row %1 \' +
'this is written on row %2 \'+
'prove #1##', value1, value2, value3)
```

% = prints every characters
# = prints a characters for each symbol # (in our example there are two symbols #)
\ = new line

[ok:=] CONFIRM ('Confirm?'[,default][,value1],..)

Shows a confirm form.
This function returns the value true if confirm button was pushed. "Default" indicates which button must be activated by default (use true to activate the "confirm" button or "false" to activate the "cancel button".

ERROR('Message'[,value1,..])

Shows an "error form" and, once pushed the button, ends the execution: transactions not confirmed by "COMMIT" will be cancelled.

[search_type:=] record.ASCENDING([ascending])

Returns or sets search order: it may be ascending (ascending:=true) or descending (ascending:=false).

record.RESET

Clears all filters, sets the primary key.

record.SETCURRENTKEY(fieldname,..,fieldname)

Sets the access key for the table. You can indicate one or more fields identifying the key; the system searches among keys of the table and selects the first key in which there are those fields. If we do not use this function the system uses the primary key.

record.LOCKTABLE

Does not allow write accesses from other transactions.

record.CALCSUMS(field#1 [,field#2]...)

---

[1] This is a quick reference to the C/Side Application Language (a language based on PASCAL syntax) used by NAVISION©. I speak only about specific functions.

Adds field values for all records filtered. All fields must be decimal and must be indicate in SumIndexField of the key used to access to the table. Value is returned in record.field#1 (record.field#2 and so on).

record.SETRANGE(fieldname[,from_value#1[,to_value])

Sets or clears (if no value is passed) a filter range (extremes included) for the field. If we indicate only from_value then the system considers all the records whose fieldname is equal to from_value. It is better use this function and not setfilter. Optimisation: for fields belonging to the key, you have to set filters according to the order they have in the key declaration.

record.SETFILTER(field, criteria)

Sets filter for the field.
Criteria is a string in which there are one or more of following symbols:

..	range
=	equal (default)
&	AND
|	OR
>	greater than
<	less than
<>	NOT =
<=	equal or less than
>=	equal or greater than

It is possible to use the following syntax:
record.SETFILTER(field, criteria,value1[,value2])
Example: record.SETFILTER(nome,'%1',TRUE)

string:= record.GETFILTER([field])

Returns filters set for the specified field or for all fields.

var := record.GETRANGEMAX(field)

Returns upper limit for the field; you will get an error if no upper limit was set.

var := record.GETRANGEMIN(field)

It functions like GETRANGEMAX, but for lower limit.

[ok:=] record.CHANGECOMPANY(CompanyName)

All operation on record will refer to the specified company and not the current one.

[ok:=] record.DELETE([RunTrigger])

Deletes the record and if the flag RunTrigger is set to true (default value is false) for first the system executes the code written for the trigger OnDelete of this table. It deletes the record whose key is equal to the one currently set.

[ok:=] record.DELETEALL([RunTrigger])

Like DELETE, but all filtered records will be deleted.

[ok:=] record.INSERT([RunTrigger])

Adds the record in the table.

[ok:=] record.MODIFY([RunTrigger])

Update the record in the table.

[ok:=] record.MODIFYALL(field, new_value [,RunTrigger])

Sets the field to new_value and updates all filtered records.

[ok:=] record.FIND(criteria)

Searches for the record according to criteria. Filters affect this function.

Criteria:
- first record;
+ last record;
< less than current key value
> greater than current key value
= equal to the key value (default)

[number:=] record.NEXT([step])

Returns the number of records following (if step >0) or preceding (if step <0) current record. Filters affect this function.

[number:=] record.COUNT

Returns the number of records in the table. Filters affect this function.

form|report|dataport.SETTABLEVIEW(from_record)

Sets filters for the record on which is based the form (or report, or dataport) according those set for from_record.

[ok:=] record.GET(value_of_the_field1_of_key [,..])

Searches for the records according to the values set for key fields..

record.INIT

Assigns default values to fields, but not those of primary key.

Field type	default value
Boolean	False
Option	0
Integer	0
Decimal	0,0
Date	0D
Time	0T
String	''

COMMIT

Ends the current transaction. When the system executes a "code unit" it enables the "write transaction"; once execution is correctly completed then the system ends the "write transaction". The COMMIT allows programmer to establish when ending a transaction.

COMPANYNAME

Returns the name of opened company.

USERID

Returns user identification.

Example

```
//This is a comment. This program ask user to give a confirmation before
//changing ZIP value in all records in which it is 99100.
//The new value is 90100.

"ZIP codes".RESET;
"ZIP codes".SETFILTER(cap,'%1','99100')

IF "ZIP codes".FIND('-') THEN BEGIN
   REPEAT
     IF CONFIRM('Do you want to chande ZIP code for country %1 ZIP %2',
        false,"ZIP codes".country, "ZIP codes".zip)
        THEN BEGIN
             "ZIP codes".zip := 95100;
             "ZIP codes".MODIFY(TRUE);
```

```
        END;
    UNTIL "ZIP codes".NEXT = 0;
```

Note: string constants are delimited by <'> while the symbol <"> is used to delimit variable names containing spaces or other particular symbols.